Sourcegraph

# Optimizing developer onboarding:

**Lessons learned and secrets to success from 31 companies and open source projects**

## Intro

Less than one week. According to a survey we just ran, that's how little time it takes for 87% of open source maintainers' individual contributors to make their first contribution. And how do private companies compare? About half of our respondents report that time to first contribution takes as much as twice or four times as long, at 2 to 4 weeks.

In this article, we'll dig into the challenges open source projects and private companies face onboarding new developers and the best practices you can use to onboard new developers more effectively. We'll examine what open source and corporate maintainers can learn from each other and why it is important to make sure collaborators are set up for success right from the beginning.

To quantify the challenges of onboarding developers to a codebase, we surveyed 31 project leads and open source maintainers who are regularly adding new people and projects to their teams. We've also run a poll with individual contributors to gather insights about common issues and pain points they face when onboarding into new projects.

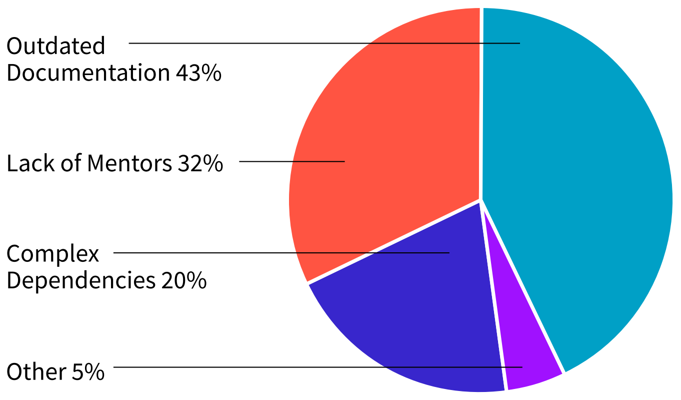
## Challenges faced with developer onboarding

Developer onboarding has the ultimate goal of turning a newcomer into an active and effective contributor to a software project, whether the project has an open source or private codebase.

Although developer onboarding might seem like a trivial subject or a process that happens organically, it can be challenging to provide developers with everything they need to be successful quickly without disrupting a team or project.

Throughout the research about developer onboarding we have conducted, it became clear that documentation is the foundational piece of any successful developer onboarding program, and keeping docs comprehensive and up-to-date can be challenging. In a recent poll we ran on Twitter, about 43% of the 401 respondents chose "Outdated documentation" as the number one pain point of developer onboarding. There is often a sizeable amount of undocumented knowledge about how things work and what is required to have them work right, both in open source as well as in companies. With increased cross-repository interdependence and complex CI/CD tooling, most codebases can't keep up with the documentation depth required to cover all edge cases their newcomers may face.

### Challenges faced when onboarding



"Lack of Mentors" came in second place in our Twitter poll with about 32% of votes. Mentoring requires time and a conscious effort to get someone up to speed, and not all teams or projects have the required resources to spare on that front. Although at first this looks like a resources issue, it may also indicate lack of comprehensive and up-to-date documentation and/or effective tools for self-served onboarding.

Our Twitter poll surfaced "Complex dependencies" in third place with 20% of votes. This refers both to complexity in setting up a proper development environment to get up and running fast, as well as having error-prone and manual deployment processes that are extremely hard to debug. In addition to making developers lose precious time and get frustrated, these will inadvertently require direct intervention from more senior members of the team.

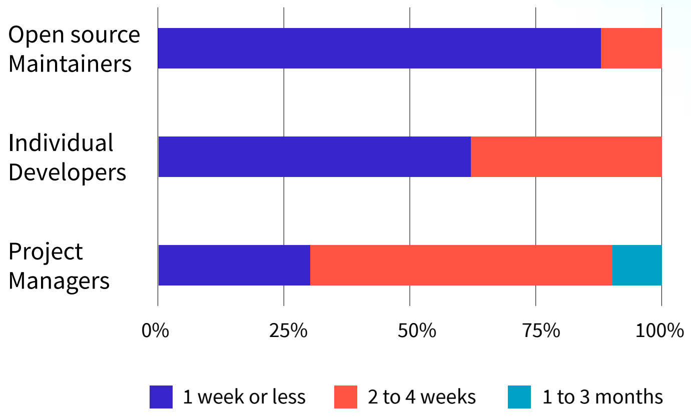
Since remote work is the new normal, it is more important than ever to establish a successful developer onboarding process that takes into account the project model, the team bandwidth, and the individual contributor needs. Considering how much developer onboarding relies on other teammates and mentors, as revealed by the survey results, it is crucial that you're able to identify which parts of this process you can optimize to remove bottlenecks and improve the overall experience of new developers joining your project or team.

## How teams are handling developer onboarding

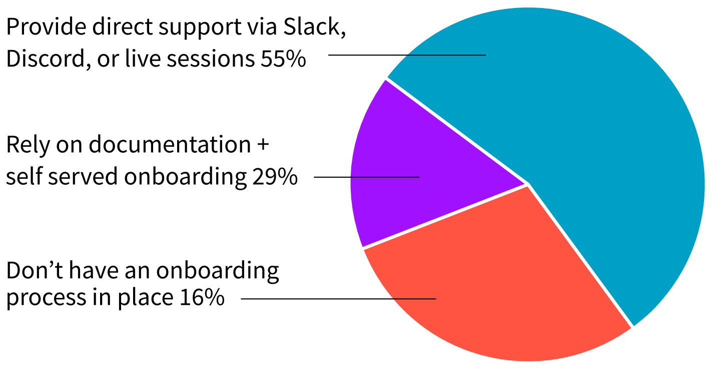
Our survey indicates that even though developer onboarding may vary greatly between different organizations, there is some consensus that having a direct communication channel with other members of the team or project is critical to a successful developer onboarding process.

About 55% of respondents reported that in addition to documentation, they also provide direct support through Discord, Slack, and live onboarding sessions. Some 29% of respondents said they rely on documentation and onboarding happens in a self-serve way. Notably, some 16% of respondents reported that they don't have a developer onboarding process in place.

### Time to first contribution



### Common approaches to developer onboarding



Open source maintainers have a lead on onboarding time, with approximately 87% of respondents reporting that individual contributors make their first contributions in less than one week. Within companies, about half of respondents reported that the time to first contribution takes 2 to 4 weeks.

The process, in general, relies on other individuals who are more experienced with the codebase and who can make themselves available as guides or mentors. Some 12% of respondents explicitly mentioned that their most valuable onboarding resource was people, and apart from IDEs and file browsers, 80% of respondents didn't use any special tooling to facilitate onboarding of new developers into their codebases.

Generally speaking, time to first contribution is a good metric to gauge the effectiveness of a developer onboarding program. Time to first contribution is a measure of how long an individual contributor takes from starting in a company or project to actually making their first code contribution to the project. According to our survey, this metric varies from 1 to 4 weeks.

## The open source ecosystem as a model for remote onboarding

The overall developer onboarding process has seen drastic change over the last few years with the normalization of remote work. Without in-person meetings or physical access to coworkers, many companies have had to reinvent their workflows and processes to better accommodate remote onboarding.

Luckily, open source organizations have been working exclusively remotely for many years, so they can serve as successful models of distributed collaboration.

Open source projects typically address remote onboarding challenges by relying more heavily on documentation and asynchronous communication. Among the open source maintainers who answered the survey, none mentioned special tools or 1:1 mentoring for onboarding, even though 62% of respondents said they have a Discord, Slack, or similar service to keep in touch with project contributors asynchronously.

## In-depth onboarding requires more time and resources

Although the perceived time to first contribution seems lower for open source projects, the commitment an open source maintainer expects from an open source contributor will differ from what a private company expects from a software developer that they pay to work on a codebase for 40 hours a week.

Long-term commitment to a project requires a more in-depth view of the codebase and its surrounding stack, which demands more time and resources spent in the process.

Companies are more likely to use synchronous communication such as live training sessions for onboarding new teammates. About 25% of respondents who self-identified as project managers or individual developers responsible for onboarding new members on the team have mentioned that they use either pair programming or live training sessions to speed up developer onboarding. Only one interviewed open source maintainer has mentioned a similar practice, highlighting that it's not something they do on a regular basis, even though it has positive effects on the productivity and engagement of returning contributors.

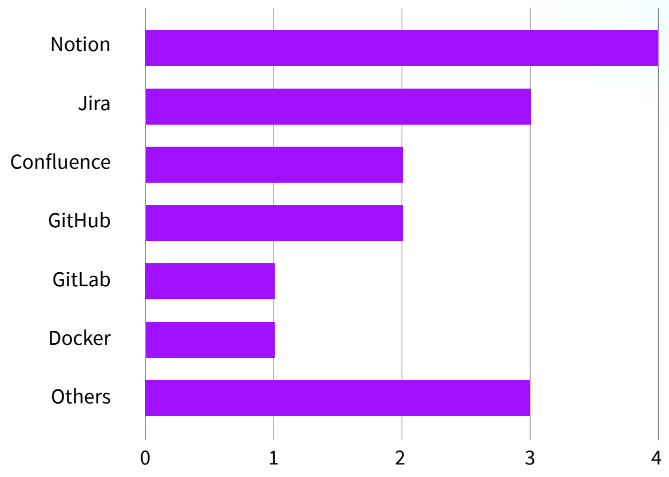
**"You can spend a month actively getting someone up to speed, or you can waste six months while you hope they figure it out for themselves."**

- quote from anonymous survey respondent.

## Developer onboarding benefits from a variety of tools and processes

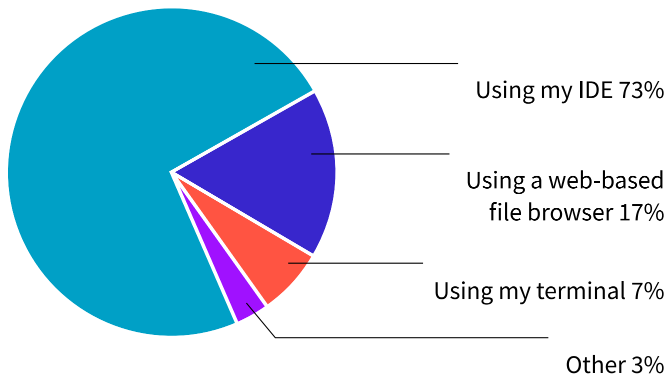
The survey suggests that the use of special tools for developer onboarding is not very common, especially among open source maintainers. Nonetheless, a few developer tools were mentioned as valuable by project managers and individual developers working for companies: Notion, Confluence, and Jira are popular choices.they do on a regular basis, even though it has positive effects on the productivity and engagement of returning contributors.

**Tool Mentions**



When walking through a code implementation in a developer onboarding meeting or presentation, 73% of respondents prefer using their own IDE, while some 13% prefer using a web-based file browser. One of the respondents mentioned that they use a live share feature in VS Code while video conferencing, which allows participants to see the code being presented in their own IDE, within their own style settings and theme. About 7% of respondents prefer to use their terminals to walk through a code implementation.

**Preferred way to present code to teammates**



In terms of processes, there is consensus that having a buddy or mentor system facilitates developer onboarding. About 24% of respondents who self-identified as members of an organization mentioned pair programming as an important practice for speeding up onboarding.

## The three pillars of a successful developer onboarding

We analyzed the answers of more than 30 individuals who are in charge of onboarding new developers both in private and open source codebases, as well as the input from interviews we conducted with project managers and open source maintainers, and that work has resulted in three best practices to incorporate into your developer onboarding process.

### 1. Excellent documentation

Having excellent documentation is a popular strategy among successful open source projects, and adopting that high standard as an integral part of your developer onboarding process should be your first concern. Relying entirely on teammates and "tribal knowledge" is not scalable and can produce an entirely different experience for each new member joining the team, especially considering time zone differences.

**"The documentation or guide should be done in such a way that it is always self-starting, clearly stating all software that needs to be installed."**
- quote from anonymous survey respondent.

Remote work requires well-designed asynchronous strategies and processes, not only to accommodate different learning styles and paces, but also to avoid bottlenecks that could be easily solved if information was more complete or easier to find. It is not good enough to have a document for everything; you must ensure that relevant information is easy to find and doesn't get buried in a forgotten Google Drive folder. It is also crucial to always keep your documentation up-to-date.

Documentation platforms such as Confluence and static documentation generators such as MkDocs are popular choices when it comes to hosting documentation. Using a service like ReadTheDocs is also a good choice, especially for open source projects. In addition to that, using a project management tool can be very helpful in documenting tickets and requests that tell an important story about the project, serving as an additional documentation source.

### 2. Optimize for early success

Whether they're joining your team or making their first pull request to your open source project, individual contributors can feel overwhelmed and frustrated when first onboarding to a new codebase. It is important to set them up for success early, and that goes beyond having good documentation. Make sure they are supported and feel comfortable asking any questions they might have.

**"Always try to make the audience feel comfortable and confident that they will be able to make part of your project. Part of onboarding is doing a good coaching job, which means making sure the newcomers will get the guidelines, etc., but also giving them the sense of being in a friendly and supportive environment."**
- quote from anonymous survey respondent.

To facilitate this process while favoring asynchronous communication, using instant messaging software such as Slack or Discord is often an excellent starting point.

Another piece of advice given by a few respondents of our survey was to start with the big picture, then dive into smaller portions of the codebase to let the new developer get acquainted with your processes and workflows. Using a good code navigation tool to present the codebase to other contributors can speed up onboarding and provide deeper insights about the project.

### 3. Create long-term connections

One of the most significant differences between developer onboarding in the context of open source software versus private organizations is the amount of resources and time made available to newcomers.

Open source maintainers tend to rely on self-serve onboarding based on documentation. Some 63% respondents who self-identified as open source maintainers offer an additional direct communication channel such as Slack or Discord for the community, but there is little live interaction. Companies, on the other hand, invest a lot more into long-term commitment, and typically have some form of direct coaching such as pair programming or live training sessions.

As an open source project gets bigger, there will be demand for continuous alignment between maintainers and individual contributors to avoid fragmentation and sustain consensus amongst the team. Although that alignment typically requires more commitment from maintainers and lead developers, offering live training sessions can be a valuable resource for retaining contributors. Live training will provide a broader understanding of the project and enable contributors to be both more productive and more impactful with their contributions.

Even though open source maintainers may have limited resources for implementing such a strategy, projects like NativeScript have successfully provided live hands-on sessions for regular contributors to the project, which seems to positively impact the project's development speed.

**"Hands-on sessions with lead maintainers can sometimes benefit regular contributors, and it would be great if we could do more of those. It helps align contributors' understanding of our general process better, which helps get their contributions merged faster and/or considered for upcoming releases. It would be nice to have a recorded session to share."**

- Nathan Walker, NativeScript maintainer

Having one or more live sessions with contributors to answer questions and walk through some key implementations can speed up the developer onboarding process and make contributors feel more confident and inspired to create long-term impact.

## Conclusion

Developer onboarding has a number of challenges that are influenced by factors such as the quality of your documentation, whether or not you offer direct contact channels for unblocking newcomers, and how much mentoring and human resources you can spare for getting new contributors up to speed.

As revealed by the results of our survey, open source projects typically take the lead on asynchronous onboarding, while companies still rely heavily on buddy systems and synchronous training. Both approaches are valid and important, since more in-depth onboarding for long-term involvement requires a better understanding of the system and having a mentor can greatly facilitate that process.

If you're working towards establishing a more formal developer onboarding process for your project or organization, you should take the following into consideration:

- Provide excellent documentation so you can set the groundwork for newcomers to get settled in while avoiding tribal knowledge and bottlenecks that could have been easily avoided if the project was better documented.
- Offer direct communication channels where individual contributors feel welcome and supported to ask any questions they might have, favoring asynchronous first so more people can join the conversation.
- Look for tools that can help with communication, project management, issue tracking, and code navigation. These tools can be used to improve visibility in the project and between collaborators, and to provide quick paths and shortcuts to important information.
- Invest in long-term connections. If you are an open source project maintainer, that typically means offering more direct access to your know-how and the project's high-level overview, which in turn helps contributors better understand the big picture and get aligned to your vision. If you are an organization, mentoring should be an integral part of your onboarding process, given that you have provided all the asynchronous resources first to set newcomers up for success at their own pace.

Be aware that developer onboarding should be a continuous process that walks side by side with a developer's journey in a company or organization. Check out our continuous development onboarding guide for more insights on this process and its benefits.

Sourcegraph

about.sourcegraph.com

Sourcegraph is a code intelligence platform that unlocks developer velocity by helping engineering teams understand, fix, and automate across their entire codebase.